AMENDMENTS TO THE CLAIMS

(IN FORMAT COMPLIANT WITH THE REVISED 37 CFR 1.121)

Please cancel claim 1 without prejudice.

1. (CANCELED)

2. (CURRENTLY AMENDED) The method of claim ~~1~~ 4, further comprising the step of:

setting said first status to said invalid state in response to a conditional store for said first operand data in said first register.

3. (CURRENTLY AMENDED) The method of claim ~~1~~ 4, further comprising the step of:

setting said first status to said invalid state in response to said first register receiving a ~~second~~ third operand data from a second register, said second register having a second status in said invalid state prior to transferring said ~~second~~ third operand data to said first register.

4. (PREVIOUSLY PRESENTED) A method of recovering from invalid data in a first register within a pipelined processor, comprising the steps of:

(A) setting a first status for said first register to an invalid state in response to a first operand data in said first register being invalid;

(B) stalling said processor in response to both (i) an instruction requiring said first operand data and (ii) said first status being in said invalid state;

(C) obtaining a second operand data from a memory in response to a conditional store for said first operand data in said first register; and

(D) stalling said processor in response to completing said obtaining to enable said second operand data to be written in said first register.

5. (PREVIOUSLY PRESENTED) The method of claim 4, further comprising the step of:

stalling said processor prior to obtaining said second operand data to prevent a third operand data from being written in said first register before said second operand data is written in said first register.

6. (CURRENTLY AMENDED) The method of claim ~~1~~ 4, further comprising the step of:

buffering said first status as a plurality of bits to provide for multiple conditions that would indicate said invalid state.

7. (CURRENTLY AMENDED) The method of claim ~~1~~ 16, further comprising the steps of:

~~setting said first status to said invalid state in response to at least one of (i) a conditional store for said first operand data in said first register, (ii) receiving a second operand data from a second register having a second status in said invalid state and (iii) a miss for a data cache read of a third operand data to be written in said first register;~~

obtaining said third operand data from ~~a~~ said memory in response to ~~at least one of said conditional store and~~ said miss; and

stalling said processor in response to obtaining said third operand data to enable said third operand data to be written in said first register.

8. (CURRENTLY AMENDED) A pipelined processor comprising:

a first register configured to buffer (i) a first operand data and (ii) a first status unique to said first operand data; ~~and~~

logic configured to (i) set said first status to an invalid state in response to said first operand data being invalid and (ii) stall said processor in response to both (a) an instruction requiring said first operand data and (b) said first status being in said invalid state; and

a bus interface unit configured to obtain a second operand data from a memory in response to a conditional store for

said first operand data in said first register, wherein said logic is further configured to stall said processor in response to completing said obtain to enable said second operand data to be written in said first register.

9. (PREVIOUSLY PRESENTED) The pipelined processor of claim 8, wherein said logic is further configured to set said first status to said invalid state in response to a conditional store for said first operand data in said first register.

10. (CURRENTLY AMENDED) The pipelined processor of claim 8, further comprising a second register, wherein said logic is further configured to set said first status to said invalid state in response to said first register receiving a ~~second~~ third operand data from said second register, said second register having a second status in said invalid state prior to transferring said ~~second~~ third operand data to said first register.

11. (PREVIOUSLY PRESENTED) A pipelined processor comprising:

a first register configured to buffer (i) a first operand data and (ii) a first status;

logic configured to (i) set said first status to an invalid state in response to said first operand data being invalid and (ii) stall said processor in response to both (a) an

instruction requiring said first operand data and (b) said first status being in said invalid state; and

a bus interface unit configured to obtain a second operand data from a memory in response to a miss for a read from a data cache of said second operand data to be written in said first register, wherein said logic is further configured to (i) stall said processor in response to completing said obtaining of said second operand data and (ii) write said second operand data in said first register in response to stalling said processor.

12. (PREVIOUSLY PRESENTED) The pipelined processor of claim 11, wherein said logic is further configured to stall said processor while said bus interface unit is said obtaining said second operand data to prevent a third operand data from being written in said first register.

13. (PREVIOUSLY PRESENTED) The pipelined processor of claim 8, wherein said first status comprises a plurality of bits to provide for multiple conditions that would indicate said invalid state.

14. (CURRENTLY AMENDED) The pipelined processor of claim 8, ~~further comprising:~~

~~a second register; and~~

~~a bus interface unit configured to obtain a third operand data for said first register from a memory,~~ wherein said logic is

further configured to ~~(i) set said first status to said invalid state in response to a conditional store for said first operand data in said first register, (ii) set said first status to said invalid state in response to receiving a second operand data from said second register having a second status in said invalid state, (iii) stall said processor in response to obtaining said third operand data for said first register, (iv) write said third operand data in said first register in response to stalling said processor and (v)~~ set said first status to said invalid state in response to a cache load-miss for said first register.

15. (CURRENTLY AMENDED) A pipelined processor comprising:

means for buffering a first operand data;

means for buffering a first status unique to said first operand data;

means for setting said first status to an invalid state in response to said first operand data being invalid; ~~and~~

means for stalling said processor in response to both (i) an instruction requiring said first operand data and (ii) said first status being in said invalid state;

means for obtaining a second operand data from a memory in response to a miss for a read of said second operand data from a data cache; and

means for stalling said processor in response to completing said obtaining to enable said second operand data to be written in said means for buffering said first operand data.

16. (CURRENTLY AMENDED) The method of claim ~~1~~ 4, further comprising the step of:

setting said first status to said invalid state in response to a miss for a read from a data cache of a ~~second~~ third operand data to be written in said first register.

17. (PREVIOUSLY PRESENTED) A method of recovering from invalid data in a first register within a pipelined processor, comprising the steps of:

(A) setting a first status for said first register to an invalid state in response to a first operand data in said first register being invalid;

(B) stalling said processor in response to both (i) an instruction requiring said first operand data and (ii) said first status being in said invalid state;

(C) obtaining a second operand data from a memory in response to a miss for a read of said second operand data from a data cache; and

(D) stalling said processor in response to completing said obtaining to enable said second operand data to be written in said first register.

18. (PREVIOUSLY PRESENTED) The method of claim 17, further comprising the step of:

stalling said processor prior to obtaining said second operand data to prevent a third operand data from being written in said first register before said second operand data is written in said first register.

19. (PREVIOUSLY PRESENTED) The pipeline processor of claim 13, wherein said logic comprises a first logic gate configured to combine said bits of said first status read from said first register.

20. (PREVIOUSLY PRESENTED) The pipeline processor of claim 8, wherein said logic comprises a first logic gate configured to combine a plurality of bits to form said first status written to said first register.

21. (CURRENTLY AMENDED) The method according to claim ~~1~~ 4, further comprising the step of:

continuing operation in a stage of said pipelined processor containing said first operand data in response to no instruction requiring said first operand data while said first status is in said invalid state.

22. (PREVIOUSLY PRESENTED) The pipelined processor of claim 8, wherein said pipelined processor is configured to continue

operating in a stage containing said first operand data in response to no instruction requiring said first operand data while said first status is in said invalid state.

23. (PREVIOUSLY PRESENTED) The pipelined processor of claim 12, wherein said logic is further configured to write said third operand data in said first register in response to writing said second operand data in said first register.